Amendments to the Specification

Please replace paragraph [0009] with the following amended paragraph:

[0009] Most common synchronization algorithms, such as those described above, ~~provide mechanisms for dealing with failed or stopped computing devices~~ are sufficiently robust that the failure of a computing device that is part of the distributed computing environment does not materially affect the synchronization of other computing devices. ~~Generally~~ Instead, such failures only result in a delay~~, but do not otherwise effect the synchronization process~~ that slows down, but does not otherwise affect, the synchronization. However, a malicious process can disrupt the above referenced mechanisms. For example, a process that indicates ~~one time to one computing device, and then a different time to another computing device,~~ widely varying times to different computing devices at approximately the same time introduces an error that can be very difficult to react to. Such malicious processes are known in the art as Byzantine failures, and mechanisms to synchronize clocks in the presence of Byzantine failures can be ~~quite complex~~ unscalable, and thus unsuitable for large-scale systems.

Please replace paragraph [0010] with the following amended paragraph:

[0010] A number of methods for synchronizing clocks in the presence of Byzantine failures are presented in the paper entitled "Byzantine Clock Synchronization" by Leslie Lamport and P. M. Melliar-Smith, dated 1984, the disclosure of which is hereby incorporated by reference, in its entirety, into the present application. One Byzantine clock synchronization algorithm reads the value of every clock in the distributed computing environment and then synchronizes its own clock to the average of the values, with the exception that if any clock differed from clock to be synchronized by more than a given amount, that clock value was replaced with the value of the clock to be synchronized. Another Byzantine clock synchronization algorithm relies on the property that in any system having a maximum of f failures, malicious or otherwise, any group of f+1 messages containing the same value must be true and a group of 2f+1 messages can be used to prove the truthfulness of the messages to a subsequent recipient. Therefore, the clock

synchronization algorithm seeks to obtain either 2f+1 messages by having a computing device send the current value of its clock to every other computing device, and then having those ~~every~~ other computing devices relay the value ~~on~~to every computing device, or seeks to obtain f+1 signed messages by having a computing device sign the current value of its clock when it is sent to every other computing device and then having ~~every~~ those other computing devices sign the signed message and relay it ~~on~~to every computing device.

Please replace paragraph [0011] with the following amended paragraph:

[0011] Unfortunately, even Byzantine-fault-tolerant algorithms, such as those described above, cannot perfectly synchronize the clocks of the computing devices that comprise the distributed computing system. For example, in order to synchronize their clocks, the computing devices will have to read each other's clocks. Reading a clock is a finite operation, which can involve multiple steps, including determining an appropriate location in memory, and reading the value out of that location in memory. Such delays can affect the accuracy of the synchronization. Another source of error can be the time required to transmit synchronization messages across a communications medium, such as a wired or wireless network. Consequently, even the most robust of algorithms cannot perfectly synchronize a ~~series~~ set of clocks.

Please replace paragraph [0012] with the following amended paragraph:

[0012] However, for all but the most time-critical applications, clock accuracy within a range can be an acceptable alternative to a perfectly synchronized series of clocks. For example, if one of the computing devices in the distributed computing environment is allowed to edit a particular file until 9:00pm, and another computing device is allowed to edit that same file starting at 9:00pm, it is not necessary that the devices exchange editing capabilities at exactly 9:00pm. Rather, the only requirement for file integrity is that one device finish editing the file before the other device begins editing the file. However, while the clock synchronization algorithms described above seek to synchronize each clock to a particular reference time by minimizing the error, ~~but~~ none of the algorithms quantify the error. Consequently, a computing device may

believe its clock is synchronized to the reference time, but it ~~is not aware of how accurate that synchronization is~~ does not have strong bounds on accuracy.

Please replace paragraph [0013] with the following amended paragraph:

[0013] Returning to the above example, if the clock of the device editing the file is a few fractions of a second slow, and the clock of the device waiting to edit the file is a few fractions of a second fast, the second device will believe that it is 9:00pm before the first device does so, resulting in two devices editing the file at the same time, and possibly corrupting the file's data. Since it is generally not important that the devices exchange editing capabilities at exactly the same time, the system would have been better served if each device had added a mere one second buffer, such that the first device ended its editing a second early, and the second device began editing a second late. The one second is not likely to impact the devices' abilities to complete their editing tasks, and yet that same second ~~can greatly reduce the possibility that the devices attempt to edit the file at the same time, and thereby corrupt it~~ is sufficient to prevent the devices from editing the same file at the same time, and thereby corrupting it. As can be seen, in many applications it is more important for each computing device to know with certainty a bound around the reference time than it is for the computing device to have its clock set to ~~the~~ a scalar reference time without the guarantee of 100% accuracy.

Please replace paragraph [0020] with the following amended paragraph:

[0020] In a further embodiment, ~~optimal~~ more efficient paths between the lowest and highest tiers of the tree of networked computing devices can be used for the propagation of messages providing the upper and lower bounds of the reference clock. Each device can use multiple paths for propagating messages up and down the tree, and can discard those paths that yield the largest deviation between the upper and lower bounds. While discarding ~~non-optimal~~ inefficient paths, each device can continue to try new paths such that the determination of the upper and lower bounds is ~~a product of~~ based on multiple paths, and not ~~subject to~~ merely a single path.

Please replace paragraph [0028] with the following amended paragraph:

[0028] For many of the functions performed by distributed computing systems, it is not necessary ~~to~~ for every device of the distributed computing system to have a precisely synchronized clock. Instead, a precise quantification of the clock synchronization error can often yield a more useful benchmark. For example, a common function performed by distributed computing systems is the maintenance of multiple copies of a database. To avoid corrupting the data stored within the database, generally only a single computing device from the distributed computing system will be allowed to modify data in the database at a given time. One common mechanism for ensuring that only one computing device is allowed to modify data at any given time is to grant leases to individual computing devices, whereby only those computing devices that have a lease have the right to edit the data in the database while their lease is pending.

Please replace paragraph [0029] with the following amended paragraph:

[0029] Often a lease granted to one computing device will expire at the same time as a lease granted to a different computing device begins. Therefore, to avoid multiple computing devices editing the same data at the same time, the clocks of the computing devices can be synchronized, such that each computing device independently maintains the same reference time, and can independently begin or end its editing of the data. However, various errors are present in even the most robust of clock synchronization algorithms. For example, the time taken to physically read the reference clock data can introduce error into a clock synchronization. Similarly, the time taken to transmit the reference clock data across a network connection can also introduce error into a clock synchronization algorithm. Furthermore, even if the clocks of two or more computing devices can be perfectly synchronized, each clock may determine the passage of time slightly differently, resulting in each clock drifting away from the synchronized time. Thus, even accurate clock synchronization algorithms cannot perfectly synchronize the clocks of two or more computing devices.

Please replace paragraph [0034] with the following amended paragraph:

[0034] Using the above determined boundaries, the first computing device can determine that its clock is somewhere between three hours and five minutes slow and two hours and 55 minutes slow. Averaging those two boundaries results in a determination that the first computing device's clock is approximately three hours slow, with definite boundaries of five minutes on either side. The first computing device can therefore reset its clock three hours ahead and can maintain the ~~data~~ knowledge that there is an upper and lower bound of five minutes in either direction. Thus, if the first computing device's lease expired at 9:00pm, the first computing device could complete its modifications by 8:55pm, by its clock, since given the five minute boundaries, 8:55pm by its clock is the earliest that the reference time can be 9:00pm.

Please replace paragraph [0043] with the following amended paragraph:

[0043] In the description that follows, the invention will be described with reference to acts and symbolic representations of operations that are performed by one or more computing devices, unless indicated otherwise. As such, it will be understood that such acts and operations, which are at times referred to as being computer-executed, include the manipulation, by the processing unit of the computer, of electrical signals representing data in a structured form. This manipulation transforms the data or maintains it at locations in the memory system of the computing device, which reconfigures or otherwise alters the operation of the computing device in a manner well understood by those skilled in the art. The data structures where data is maintained are physical locations of ~~the~~ memory that have particular properties defined by the format of the data. However, while the invention is being described in the foregoing context, it is not meant to be limiting as those of skill in the art will appreciate that various acts and operations described hereinafter may also be implemented in hardware.

Please replace paragraph [0045] with the following amended paragraph:

[0045] As described above, a computing device can determine an upper and lower boundary for a reference time by sending a request to a reference time ~~computing device~~ source, storing

the time (according to the sending device's clock) when the request was sent, receiving a response containing the reference time from ~~a~~ the reference time computing ~~device~~ source, and storing the time (according to the receiving device's clock) when the response was received. The reference time source can be a single reference time computing device, as shown in Figure 2, or two or more networked reference time computing devices acting as a single unit such as in the manner to be described further below. Figure 2 illustrates a request packet 231 sent by computing device 221. As will be clear to those skilled in the art, the present description uses the term "packet" to describe any unit of data which can perform the functions enumerated, and does not intend to limit the term "packet" to mean only a "network packet" or similar quanta defined by network protocols. The computing device 221 can send the request packet 231 to a computing device on a higher level, such as devices 211, 212, or 213. In the illustrative example of Figure 2, the request packet 231 is sent to computing device 211, which can also add its own request packet and create a new combination request packet 232. The computing device 211 can then send packet 232 to a computing device on a higher level. In the illustrative example of Figure 2, the higher level contains the reference time computing device 201, which, because it is acting as the reference time source, can receive packet 232, encode the reference time, and create a new packet 233. The new packet 233 can comprise the reference time and the particular request packets to which the reference time computing device 201 is responding. Because one of the bounds of the reference time is determined by when the request packet was sent, the computing device requesting the reference time, such as computing devices 211 or 221, can either delay sending another request before receiving a response to a previous request, or a mechanism for linking requests to responses can be used. To avoid unnecessary delays, the latter option is preferable. Therefore, as shown in Figure 2, the response packet 233 comprises the request packets to which it responds so that the receiving computing devices 211 and 221 can determine the bounds appropriately.

Please replace paragraph [0048] with the following amended paragraph:

[0048] The operation of device 221 can occur in a manner similar to that of device 211, described above. Specifically, device 221 can receive packet 234 from device 211 at 8:05pm by the clock of device 221. Upon extracting the reference time of 9:00pm encoded in packet ~~233~~234, the computing device 221 can determine the following bounds: the reference time of 9:00pm cannot have occurred prior to 7:55pm by the clock of device 221, since the device 221 had not yet even sent the request prior to that time, and the reference time of 9:00pm cannot have occurred later than 8:05pm by the clock of device 221, since by that time the device 221 had already received the packet 234. Consequently, the computing device 221 can determine that its clock is between 55 minutes and an hour and five minutes slow. It can, therefore, set its clock ahead by one hour, and can then store the boundary information that its clock is set to the reference time with an error of plus or minus five minutes. Again, as explained in detail above, the device can also maintain an earliest time and a latest time, which in the present example would be 8:55pm and 9:05pm. As can be seen, because device 221 is further down the illustrative network tree 200, the bounds of the reference time that it determines may be greater than the bounds of the reference time determined by devices at higher levels of the network tree 200.

Please replace paragraph [0049] with the following amended paragraph:

[0049] However, the above illustration assumes that the distributed computing environment does not contain any malicious devices or processes. For example, the above illustration assumes that device 211 does not inappropriately modify the packets that it receives. If, however, device 211 was malicious, it could modify the response packet 234 in such a manner that device 221 treats the packet 234 as a response to a request other than the request in packet 231. In such a case, the bounds determined by device 221 ~~might~~ would be incorrect, ~~in the sense that they might be~~ as they would either be too large or too small. Alternatively, ~~T~~the malicious device 211 could ~~also make the bounds larger than they would need to be, but it can achieve this same goal by~~ simply not forward~~ing~~ the packet at all, in which case device 211 would obtain no

synchronization information. Both of these situations are undesirable, but the former is worse because it can lead to data corruption or inconsistency.

Please replace paragraph [0050] with the following amended paragraph:

[0050] To provide a fault tolerant mechanism for quantifying the synchronization of a device's clock to a reference time, one embodiment of the present invention contemplates the use of cryptographic algorithms to protect the data sent by the various devices. Turning to Figure 3a, a distributed computing system in the form of a network tree diagram 300 is shown. The distributed computing system contains computing devices 311, 312, and 313 at one level, and devices 321, 322, 323, 324, 325, 326, and 327 at a lower level. The reference time computing device 301 is shown as a member of a Byzantine fault tolerant system 303, together with reference time computing device 302. In the illustrated distributed computing system, the Byzantine fault tolerant system 303, as a whole, acts as the reference time source, and none of the individual members of the system 303, such as reference time computing devices 301 or 302 can, by themselves, act as the reference time source. The operation of the Byzantine fault tolerant system 303 will be explained in detail below.

Please replace paragraph [0052] with the following amended paragraph:

[0052] Each device in the network diagram 300 can use a similar system to protect against malicious devices. For example, when device 311 receives the nonce data in packet 331, it can add its own nonce and then cryptographically ~~encrypt~~ hash the two nonces into a packet, such as by using ~~a~~ known hashing algorithms. Similarly, when the reference time computing device 301 receives the hashed nonces from device 311, it can, together with the other reference time computing devices that comprise the Byzantine fault tolerant system 303 that is the reference time source, add the ~~time~~ reference time information and ~~hash~~ sign the combination of the reference time and the hashed nonces from device 311 and return that to device 311.

Please replace paragraph [0053] with the following amended paragraph:

[0053] Thus far, the illustrations described have only traced the path of a request originating from a single computing device. However, a higher level computing device, such as device 311 may have multiple lower level computing devices connected to it, with each device seeking to synchronize its clock. One embodiment contemplated by the present invention is for the higher level devices to simply send along a request, in the manner described above with reference to Figure 2, each time they receive a request from any lower level device. As will be evident to those skilled in the art, such a mechanism can quickly inundate higher level devices that may have hundreds of devices connected to them through multiple lower level layers. While sending a lot of messages may not noticeably impact the performance of the overall distributed computing system when using a simple mechanism, such as that illustrated in Figure 2, the use of cryptographic signatures as shown in Figure 3a can markedly increase the computing cost of creating and sending a message. Consequently, it may not be desirable for each higher level computing device to hash and send along messages each time they arrive from lower level computing devices.

Please replace paragraph [0054] with the following amended paragraph:

[0054] An alternative approach contemplated by an embodiment of the present invention allows the higher level computing devices to collect some or all of the messages from lower level devices prior to hashing those messages and transmitting them onto to higher level devices. Returning to Figure 3a, computing device 311 can wait to receive, not only the nonce 331 from computing device 321, but also the nonce 332 from computing device 322, and the nonce 333 from computing device 323. Once the computing device 311 has received each of these nonces, it can create its own nonce, and then hash this collection of four nonces together into a packet 334 for transmission to the higher level computing device 301. The illustrated packet 334 shown in Figure 3a ~~includes a box~~ is marked with an "H" to indicate that ~~the~~ its contents ~~of that box~~, in this case the nonces from devices 321, 322, 323, and 311, are hashed together.

Please replace paragraph [0055] with the following amended paragraph:

[0055] By waiting until some or all of the requests from the lower level devices have received before transmitting a request to higher level devices, a computing device can greatly reduce the number of messages transmitted through the distributed computing system 300, and can reduce the computational expense of computing many cryptographic hashes. The reference time computing devices that comprise the Byzantine fault tolerant system 303 that is the reference time source, including devices 301 and 302, can similarly wait to receive requests from some or all of the devices in the lower levels before initiating the encoding of a reference time and the transmitting ~~the~~ of responses ~~back down the network tree~~ in accordance with the particular Byzantine fault tolerant algorithm implemented by the system 303, which will be described in further detail below.

Please replace paragraph [0056] with the following amended paragraph:

[0056] Turning to Figure 3b, the distribution of the response of the reference time ~~computing devices~~ source is shown. As will be described in more detail below, Figure 3b illustrates a response from a Byzantine fault tolerant system 303, which is the reference time source, and which comprises ~~comprising~~ reference time computing devices 301 and 302. While a Byzantine fault tolerant system is one ~~mechanism~~ reference time source contemplated by one embodiment of the present invention for ensuring that a malicious device or process does not disseminate an incorrect reference time, other ~~mechanisms~~ types of reference time sources are also contemplated by other embodiments of the present invention. For example, a reference time ~~computing device 301~~source can be a closely monitored computing device containing the de facto standard reference time. One example of such a device is the computing device operating the atomic clock in Boulder, Colorado, which is the reference used by many military and industrial applications in the United States. Alternatively, a reference time ~~computing device 301, alone or in combination with other reference time computing devices, such as device 302,~~source can implement a known fault tolerant system for ensuring a correct reference time. Therefore, as can

be seen, the mechanism used to ensure a correct, uncorrupted reference time is not essential to embodiments of the present invention.

Please replace paragraph [0057] with the following amended paragraph:

[0057] For simplicity, an initial description of the operation of one cryptographic response mechanism according to an embodiment of the invention will be presented without reference to the Byzantine fault tolerant system 303 or ~~the~~ any additional reference time computing devices, such as reference time computing device 302, shown in Figure 3b. Thus, the initial description will assume that the reference time computing device 301, by itself, acts as the reference time source ~~maintains a proper reference time~~ and that it is operating properly and is not malicious.

Please replace paragraph [0060] with the following amended paragraph:

[0060] The use of the cryptographic mechanisms described above enables the Byzantine fault tolerant determination of the bounds of a reference time. As will be explained in detail below, malicious devices or processes cannot successfully edit or tamper with the data being transmitted. The only malicious activity which a Byzantine device could perform is to delay or drop the data being transmitted. However, such a delay or a drop may only result in the bounds of the reference time increasing to positive and negative infinity, a delay or a drop cannot result in incorrect bounds. Furthermore, because devices can use multiple paths to determine the most accurate bounds, a path that yields unreasonably large bounds will be avoided, thereby ~~minimizing~~ mitigating the impact of malicious processes or devices.

Please replace paragraph [0062] with the following amended paragraph:

[0062] The other boundary may not have been as eaffected by Byzantine faults since even a malicious device or process cannot reverse time. Thus the determination that the reference time cannot have occurred after the receipt of the reference time by the computing device, due to the causal nature of the universe, remains valid even in the presence of Byzantine failures.

Please replace paragraph [0063] with the following amended paragraph:

[0063] Finally, the cryptographic mechanisms described do allow for a Byzantine fault tolerant determination of the reference time. Thus, ~~whatever reference time was sent by the reference time computing device, such as device 301, that same reference time should be received by all of the requesting devices, even in the presence of Byzantine failures~~ all requesting devices will compute bounds on the reference clock that, in fact, include the correct reference time as determined by the reference time source.

Please replace paragraph [0064] with the following amended paragraph:

[0064] However, as described above, the reference time ~~computing device~~ source may itself be malicious, and may purposely provide incorrect reference time data. As also described above, various mechanisms can be implemented to ensure that the reference time ~~computing device~~ source is not malicious, or to account for its maliciousness such that the reference time ultimately sent is, in fact, the correct reference time. One such mechanism contemplated by the present invention is the use of a Byzantine fault tolerant system~~, comprising some or all of~~ to act as the reference time ~~computing devices~~ source~~,~~ and to determine the correct reference time to distribute to the requesting devices.

Please replace paragraph [0066] with the following amended paragraph:

[0066] In the Byzantine Paxos algorithm, a leader device initiates a first phase by requesting that the remaining devices promise to vote for a given proposal that the leader intends to submit. Each device then sends its last vote information to all of the other devices. Once any device receives 2f+1 messages, each containing the same information, that device knows that the contained information ~~is true~~ was previously agreed upon, and it has a sufficient collection of messages to prove to another device that the information ~~is true~~ was previously agreed upon. Because there are at most f failed or malicious devices, any collection of at least f+1 equivalent messages is sufficient to show that the messages ~~are true~~ contain information that was previously agreed upon, since at least one non-failed, non-malicious device's message is in that group. A

group of 2f+1 messages is sufficient to prove to another device that the messages contain information that was previously agreed upon ~~are true~~, because even if the other device suspects that f malicious devices collaborated to collect the messages, at least f+1 messages remain that are not from malicious sources. Those remaining f+1 messages are then sufficient to prove to the other device that the messages contain information that was previously agreed upon ~~are true~~, as just described.

Please replace paragraph [0067] with the following amended paragraph:

[0067] With the 2f+1 messages collected, each device can send an indication of a safe proposal number, last vote information, and the 2f+1 messages as proof ~~of the veracity of the safe proposal number~~ to the leader. Once the leader receives a quorum such messages ~~and collections~~, it can attempt to have the devices agree on a ~~value or operation for the~~ selection for a safe proposal value by starting ~~the~~ a second phase. The leader will attach the quorum of received messages as a proof of the safety of the proposal to each of the devices. Each of the devices will signal their willingness to vote for the proposal to every other device. Once a device receives a quorum of messages from other devices indicating that they will accept the proposal, the device accepts the proposal and transmits its acceptance to the leader. If the leader receives a quorum of acceptances, then the leader knows that the proposal was selected. Additional information regarding the Byzantine Paxos algorithm can be found in co-pending application Serial No. 10/184,773, filed on June 28, 2002, and entitled "Byzantine Paxos", the disclosure of which is herein incorporated by reference, in its entirety, into the present application.

Please replace paragraph [0069] with the following amended paragraph:

[0069] Yet another alternative Byzantine fault tolerant system contemplated by an embodiment of the present invention can be implemented by using a three phase algorithm instead of the two phase algorithms described above. Specifically, in a first phase, a leader sends a signed proposal to each of the other devices. Each of the other devices then enters a second phase if the proposal was signed properly and contains the proper sequence number. In the

second phase, each of the other devices sends a message to every other device indicating that it has provisionally accepted the proposal. If a device receives an additional 2f messages (in addition to its own) indicating that those other devices have provisionally accepted the proposal, the device will accept the proposal and will transmit a commitment message to each of the other devices. Once each device receives 2f+1 commitment messages, it executes the operation and returns the results to the client. If the client receives f+1 equivalent results from f+1 devices, then it knows that the result is correct. Additional information regarding this Byzantine fault tolerant algorithm can be found in the paper entitled "Practical Byzantine Fault Tolerance" by Miguel Castro and Barbara Liskov, dated February 1999, published in the Proceedings of the Third Symposium on Operating Systems Design and Implementation", the disclosure of which is hereby incorporated by reference in its entirety into the present application.

Please replace paragraph [0071] with the following amended paragraph:

[0071] Embodiments of the present invention contemplate the use of any of the above described Byzantine fault tolerant algorithms to provide a system which acts as a reference time source and which can agree upon a proper reference time even in the presence of malicious devices or processes. Returning to Figure 3b, a Byzantine fault tolerant system 303 comprising reference time computing devices 301 and 302, and additional devices not shown, can be the reference time source. Using any of the above described algorithms, the Byzantine fault tolerant system 303 can agree on a set of hashed collections that can be used to create a response packet, such as packet 340. For example, device 301 can act as a leader in the system 303 and can propose that hashed collections 334, 335, and 336 be used in formulating the response packet 340. Each device of the system 303 can then, according to the particular Byzantine fault tolerant algorithm being used, as described above, decide whether the agree with device 301's proposal.

~~be used, each of the devices can store the hashed collections and the system 303 can proceed to determine a proper reference time with which to respond.~~

Please replace paragraph [0072] with the following amended paragraph:

[0072] ~~Once again~~ In addition to proposing which hashed collections can be used in formulating the response packet 340, a leader, such as device 301, or a client, can also propose a reference time to be included in the response packet 340~~to use in responding to the hashed collections of messages~~. Each of the devices in the Byzantine fault tolerant system 303 can then determine whether to accept the ~~proposed time~~ proposal or not. One algorithm for determining whether to accept the proposal, which includes the proposed reference time, is to compare the proposed reference time to the clock of the particular voting device. If the proposed reference time is within a predetermined reliability range of the clock of the voting device, the voting device can accept the proposal. If a sufficient number of voting devices accept the proposal, in the manner described above with respect to the various Byzantine fault tolerant algorithms, then the accepted reference time can be used in the creation of packet 340. Similarly, if a sufficient number of voting devices reject the proposal, it can be an indication of a malicious or faulty leader, and the Byzantine fault tolerant system 303 can select a new leader in the manner of the Byzantine fault tolerant algorithms described above.

Please replace paragraph [0074] with the following amended paragraph:

[0074] As described above, many Byzantine fault tolerant algorithms ~~simply~~ send a collection of messages to the client, ~~with~~ where a sufficiently large collection of ~~similar~~ identical messages ~~being~~ is sufficient for the client to determine that the content of the messages is correct. Similarly, the Byzantine fault tolerant system 303 can ~~result in~~ send multiple messages ~~being sent~~ to the computing devices at the next lower level. Therefore, as shown in Figure 3b, computing device 311 can receive the packet 340, together with the hashed collections 335 and 336 from both reference time computing device 301, as message collection 341, and from reference time computing device 302, as message collection 342. Additionally, device 311 may

receive analogous messages from the other devices comprising Byzantine fault tolerant system 303. If a sufficient number of these message collections are equivalent, then the computing device 311 can determine that they contain the correct information, even in the presence of malicious processes or devices. In one embodiment, if device 311 receives f+1 equivalent message collections, then it can trust the message collection, even in the face of at most f malicious devices.

Please replace paragraph [0082] with the following amended paragraph:

[0082] As is known by those skilled in the art, most computing devices derive time information from a crystal oscillator. Such crystal oscillators have a specific frequency that can be accurately measured. However, each crystal oscillator's frequency may be ever so slightly different from any other crystal oscillator's frequency. If the cause of the drift between the clock of a computing device and the reference time computing device is due primarily to this hardware difference, then the drift is likely to be constant. In such a case historical rate information can be very useful in compensating a determined reference time bound for drift. For example, historical rate information may reveal that a given computing device's oscillator varies from the reference time computing device's oscillator by 10 milliseconds for every minute of elapsed time. Consequently, the determined reference time bound can simply be increased by 10 milliseconds for every ~~second~~ minute that has elapsed since the bound was determined.

Please replace paragraph [0083] with the following amended paragraph:

[0083] Sometimes, however, the drift between the clock of a computing device and the clock of a reference time computing device is not due to predictable, measurable properties, such as the exact oscillator frequency. Often the cause of the drift may be due to environmental factors, such as heat or vibration. In such a case, the drift is not likely to be linear, but may be modeled only though higher order equations. In one embodiment of the present invention, first order, second order, or higher order equations can relate the drift as a function of the elapsed time or other variables. An alternative embodiment of the present invention avoids the complexities of

accurately modeling what might be random processes, such as heat, and ~~simply seeks to select an~~ instead applies a boundary-based approach ~~upper bound for the drift~~. Specifically, the predictable components of the drift, such as those described above, can be removed from the measurements, leaving only the more random components of the drift, providing a more accurate boundary for the non-predictable components of the drift. Thus, the exact drift may not be easily modeled, but it can be determined with reasonable accuracy, especially if the predictable components of the drift dominate ~~that the drift is rarely greater than 100 milliseconds for every minute of elapsed time~~. As before, the determined reference time bound can modified to account for this drift ~~by increasing for 100 milliseconds in each direction the bound for every second that has elapsed since the bound was determined~~. While such a calculation may result in a broader range than would have been obtained with a more accurate model, the increase in the range may not affect many applications, while the simplicity of the calculation may ~~save valuable processor bandwidth~~ avoid unnecessary algorithmic complexities.